5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/607,397	06/30/2000	Damon Barry	13768.132	9886

47973 7590 06/12/2007

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
KISS, ERIC B	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
06/12/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/607,397
Filing Date: June 30, 2000
Appellant(s): BARRY ET AL.

F. Chad Copier (Reg. No. 54,047)
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed February 23, 2007, appealing from the Office action mailed August 15, 2006.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

"TETware User Guide, Revision 1.2," The Open Group, TET3-UG-1.2, September 18, 1998 ("TET_UG");

"Release Notes for TETware Release 3.3," The Open Group, TET3-RN-3.3, September 18, 1998 ("TET_RN"); and

"TETware Programmers Guide, Revision 1.2," The Open Group, TET3-PG-1.2, September 18, 1998 ("TET_PG").

6,505,342 HARTMANN et al. 1-2003

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1, 2, 4, 7, 10-17, and 20-28 are rejected under 35 U.S.C. 102(b) as being anticipated by the TETware Release 3.3 software product (hereinafter TETware) released September 18, 1998 by The Open Group, as evidenced by: "TETware User Guide, Revision 1.2" (hereinafter TET_UG), "Release Notes for TETware Release 3.3" (hereinafter TET_RN), and "TETware Programmers Guide, Revision 1.2" (hereinafter TET_PG).

As per claim 1, TETware is disclosed with a computer system comprising:

one or more program modules (test suite directories) storing one or more available test cases (see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed), each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface (the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG; the use of different source languages to build cases is also disclosed, e.g., C, C++, Shell, Korn Shell, or Perl; see, for example, section 2.4 of TET_UG describing the API components as linkable object code);

a harness client comprising a set of instructions that (i) receives user input specifying one or more filenames corresponding to the one or more program modules (see, for example, section

5.3.2 of TET_UG), (ii) employs the connector to scan for and discover the one or more available test cases that are stored in the one or more program modules and to organize the one or more available test cases into a test case hierarchy (see, for example, section 5.3.2 of TET_UG; test cases are organized into test suites, which are organized in test suite directories under the test suite root directory), and (iii) receives user input for indicating which of the one or more available test cases in the test case hierarchy are selected to be executed on the computer program (see, for example, section 5.3.2.2 of TET_UG; the scenario file specifies which specific test cases of a specific test suite in a specific test suite directory, relative to the test suite root directory, are to be executed);

a harness comprising a set of instructions that (i) receives the test case hierarchy, (ii) traverses the test case hierarchy, and (iii) executes each of the one or more available test cases that is selected to be executed (test scenario) on the computer program using the corresponding language and format independent interface of the selected test case to ensure that the computer program processes as intended (test case controller; see sections 2.1 and 2.2 of TET_UG; the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG);

a connector, comprising a set of instructions that (i) scans for the one or more available test cases stored in the one or more program modules, (ii) organizes the one or more available test cases into the test case hierarchy by extracting the one or more available test cases from the one or more program modules (see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed; section 2.4 of TET_UG; and section 2.4.4 of TET_PG describing the handling of non-API test cases),

and (iii) selectively integrates an interface between the test case hierarchy and the harness regardless of the language or format in which the one or more available test cases were written (test case managers and API libraries; see section 2.4 of TET_UG; see also section 2.4.4 of TET_PG describing the handling of non-API test cases);

a processor for executing each selected test case, the harness, the harness client, and the connector (inherent in the operation of the UNIX and WINDOWS operating systems used to implement TETware; see section 1.1 of TET_UG).

TETware is further disclosed with one or more test cases comprising a test suite in the hierarchy and one or more test suites comprising a test module in the hierarchy (see section 2.2 of TET_UG; see further, section 4.1 of TET_PG and 5.3.2.1 of TET_UG). When an individual scenario from the scenario file is processed, one or more test cases may be invoked (as described, for example, 4.2.4.3 of TET_PG and 5.3.2.4 of TET_UG).

As per claim 2, TETware is further disclosed with the set of instructions of the harness and the set of instructions of the connector utilizing an architecture that defines a means for accessing a resource over a network (see section 2.6.3 of TET_UG).

As per claim 4, TETware is disclosed with a method comprising:

The harness client receiving user input that (i) specifies a search property to identify one or more test cases of interest (see, for example, section 5.3.2 of TET_UG), (ii) selects one or more test cases from the one or more test cases of interest to execute on the computer program (see, for example, section 5.3.2.4 of TET_UG), and (iii) specifies how the one or more selected test cases are to be executed on the computer program (see, for example, section 5.3.2.2 of TET_UG);

the connector scanning the binary program module (test suite) storing the plurality of individually accessible test cases, for one or more test cases of interest (see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed), each test case having a language and format independent interface for executing the test case on the computer program regardless of the language or format used to develop the test case (the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG);

the connector extracting the one or more test cases of interest from the binary program module (see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed);

the connector organizing one or more test cases into a test case hierarchy (test suite structure; see section 2.2 of TET_UG; see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed);

the connector interfacing a harness with the one or more test cases of interest (see section 6.4 of TET_UG; see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed), wherein the interfacing allows the harness to recognize and execute the one or more test cases of interest regardless of the language or format in which the one or more test cases of interest were developed (test case controller; see sections 2.1 and 2.2 of TET_UG; the test cases are built and

executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG); and

the harness traversing the test case hierarchy and executing each of the one or more selected test cases to test the computer program (see the description of the test case controller beginning on page 105 of TET_UG).

TETware is further disclosed with one or more test cases comprising a test suite in the hierarchy and one or more test suites comprising a test module in the hierarchy (see section 2.2 of TET_UG; see further, section 4.1 of TET_PG and 5.3.2.1 of TET_UG). When an individual scenario from the scenario file is processed, one or more test cases may be invoked (as described, for example, 4.2.4.3 of TET_PG and 5.3.2.4 of TET_UG).

As per claim 7, TETware is further disclosed with a step of determining whether one or more of the test cases of interest are identified as being deselected, wherein a deselected test case is not executed on the computer program (see, for example, the "-n" command line option of the test case controller on page 107 of TET_UG).

As per claims 10 and 11, TETware is further disclosed with excluding test cases determined to be deselected from a selection of a test suite or scenario (see, for example, the "-n" command line option of the test case controller on page 107 of TET_UG).

As per claims 12-14, TETware is further disclosed with the step of traversing further including executing the one or more test cases on a thread pool comprising one or more threads, and further discloses testing single-threaded and multi-threaded (thread-safe) models (see section 17.4 of TET_PG).

As per claims 15-17, these are computer-readable medium versions of the method discussed above (claim 4), wherein all limitations have been addressed as set forth above. Furthermore, the use of such a computer-readable medium containing executable code is inherently necessary for the operation of the UNIX and WINDOWS operating systems used to implement TETware (see section 1.1 of TET_UG).

As per claim 20, TETware is further disclosed with user-selected (through a harness client user interface) test cases (see the description of the test case controller and command line usage beginning on page 107 of TET_UG).

As per claims 21-23, see the disclosure applied above in the rejection of claims 12-14.

As per claim 24, TETware is disclosed with a method comprising:

specifying one or more filenames for identifying one or more program modules storing one or more test cases, each comprising a set of instructions for testing a feature of the computer program through a language and format independent interface (see, for example, section 5.3.2 of TET_UG; the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG);

identifying the one or more test cases within the one or more program modules (see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed);

translating the identified one or more test cases into a test case hierarchy (a test scenario see, for example, section 2.5.2 of TET_PG, which describes "Test scenario definitions" that specify which test cases of a test suite are to be executed);

indicating that the one or more test cases in the test case hierarchy are to be executed on the computer program (see, for example, section 5.3.2 of TET_UG);

providing an interface to the test case hierarchy in order to recognize and execute the one or more test cases regardless of the language or format in which the one or more test cases were written (test case controller; see sections 2.1, 2.2, and 2.4 of TET_UG; the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG); and

running each of the one or more test cases in the test case hierarchy to test the computer program (test case managers and API libraries; see section 2.4 of TET_UG; see also section 2.4.4 of TET_PG describing the handling of non-API test cases; the test cases are built and executed, regardless of their source language, through the same test case controller; see, for example, the description of build mode in section 6.2.3 of TET_UG).

As per claims 25-27, TETware is further disclosed with executing the one or more test cases on a thread pool comprising one or more threads, and further testing single-threaded and multi-threaded (thread-safe) models (see section 17.4 of TET_PG).

As per claim 28, Applicant's specification defines "test module" as "a set of one or more test suites." (Specification p. 8, line 17.) Accordingly, as the individual test suites of TETware are contained in test suite directories, and the test suite directories are contained in the test suite root directory (TET_UG section 5.2.6; TET_PG section 2.3), it follows that either the test suite directories or the test suite root directory meet the recited "test module". Further, the filenames of individual test cases, being specified relative to the test suite root directory (and the test suite directory therein), correspond to the test modules by pointing to their contents.

Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over TETware and the associated cited documentation as applied to claim 1 above, and further in view of U.S. Patent No. 6,505,342 to Hartmann et al.

As per claim 3, TETware is disclosed with such a system (see disclosure applied above to claim 1), but is not expressly disclosed with a COM technology architecture. However, Hartmann et al. teach a system for testing components that use middleware, such as COM/DCOM (see column 2, line 61 through column 3, line 4). Therefore, it would have been obvious to one having ordinary skill in the computer art at the time the invention was made to modify the system of TETware to include a COM architecture as per the teaching of Hartmann et al. One would be motivate to do so to gain the advantage of supporting and testing implementations in a standardized object-oriented middleware.

(10) Response to Argument

TETware discloses the claimed test hierarchy.

Regarding the claimed hierarchy of test cases, test suites, and test modules, there is no dispute that the test suites of TETware comprise one or more test cases. (*See* Brief at p. 10.) Further, as appellant has previously acknowledged, within the defined TETware directory structure, test suites are organized under the test suite root directories. (*See* Remarks (02/06/2006) at p. 10 ("*As noted in Applicants' prior response, TETware discloses grouping test cases within as test suite.* [TET_PG], *section 2.2. Test suites are organized as directory hierarchies—the top of each test suite directory is known as the test suite root directory.* [TET_UG], *section 5.2.6. All files in a test suite reside below the test suite root directory (or in a specified alternate execution directory).* [TET_PG], *section 2.3;* [TET_UG], *section 5.2.6.*").)

Further, the test suite root directory is usually located immediately below the "tet root" directory. TET_PG at section 2.3. Appellant's specification defines "test module" as "a set of one or more test suites." (Specification p. 8, line 17.) Accordingly, as the individual test suites of TETware are contained in test suite root directories (TET_UG at section 5.2.6), and the test suite root directories are contained in the tet root directory (TET_PG at section 2.3; TET_UG at section 5.2.6), it follows that either the test suite root directories or the "tet root" directory meet the recited "test module". Figure 17 of "TETware Programmers Guide, Revision 1.2," included below, further illustrates an example of the test hierarchy disclosed by TETware. TET_PG at p. 148.

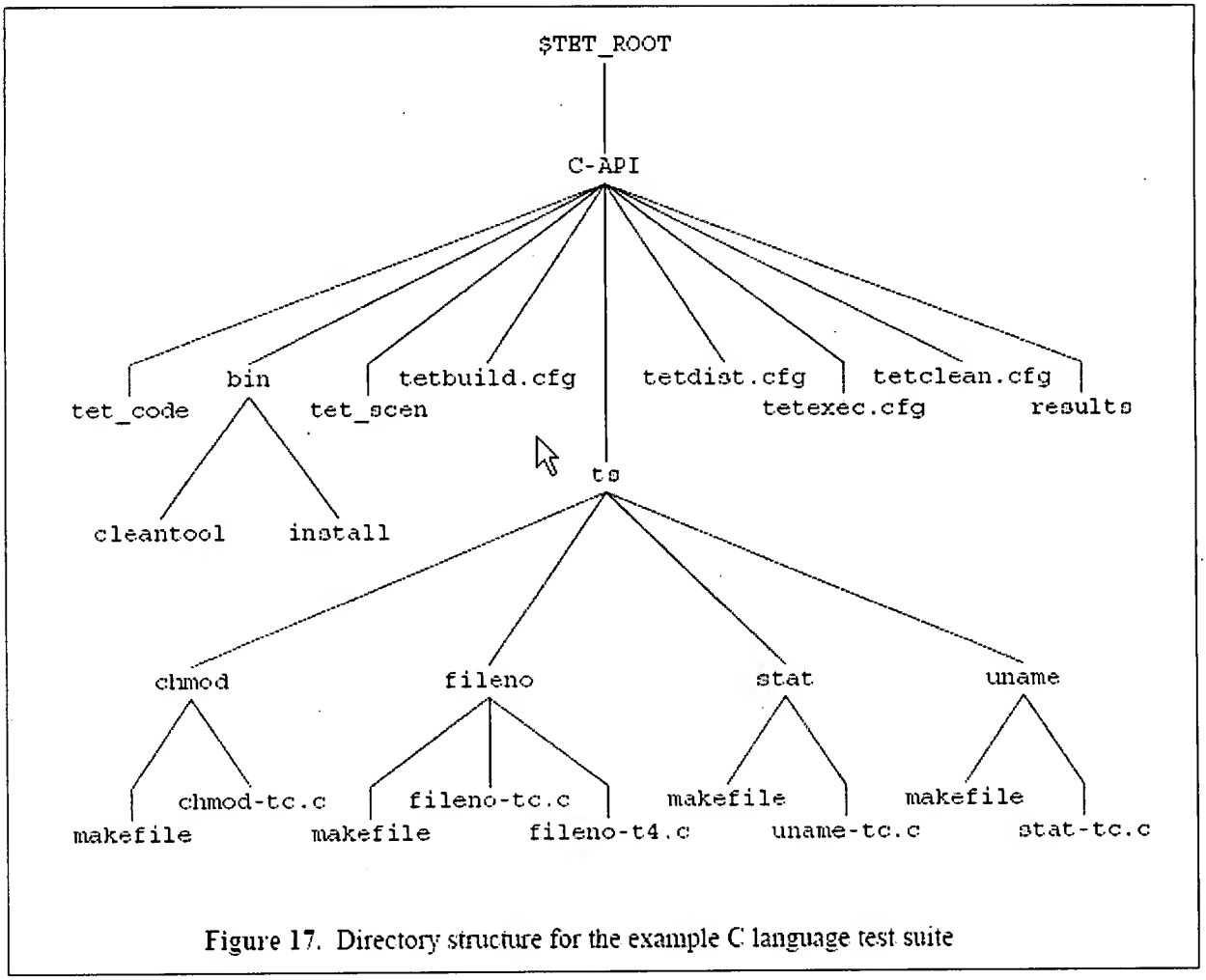

Figure 17. Directory structure for the example C language test suite

This figure illustrates the components that make up sample test suite "C-API." TET_PG at 148. The C-API test suite resides in a subdirectory under the test suite root directory, "$TET_ROOT." *Id.* at 147. Thus, $TET_ROOT may be considered the test module (a set of one or more test suites) in this example (or alternatively, the C-API test suite root directory, as a set of one test suite, may likewise be considered a test module). Within the C-API directory is a sub-directory, "ts," containing several test cases ("chmod", "fileno", "stat", and "uname"). *Id.* at 148. Appellant's claimed test hierarchy is similarly illustrated in appellant's FIG. 3.

TETware discloses the claimed test case discovery.

TETware uses a Test Case Controller (tcc) to process test suite scenarios. TET_UG at section 6.2.1. Before invoking tcc, the user is required to ensure that the value of the TET_ROOT environment variable points to the tet root directory on the local system. TET_UG at section 6.2.2. User input to accomplish this is described in several examples. *E.g.*, TET_UG at section 4.3.1. Once the tet root directory is specified, the tcc may be instructed to execute selected test cases (the user selects from the available test cases by invoking a particular test scenario) from within a test suite under the tet root directory. TET_UG at section 6.2.3 (describing execution of the "all" scenario by default or execution of a user-specified scenario (which in turn specifies particular test cases of the test suite for execution)); TET_UG at section 5.3.2 (describing the contents of the scenario file). The test scenario of TETware specifies which specific test cases of a specific test suite in a specific test suite directory, relative to the test suite root directory (and the tet root directory), are to be executed. (TET_UG at section 5.3.2.2). In order for this functionality to be realized, TETware must be able to traverse (scan and discover) files (including the available test cases) within the hierarchical directory structure.

The Rejection of Claim 3 under 35 U.S.C. § 103(a)

Appellant's arguments regarding claim 3 mere repeat the allegations of deficiencies in the TETware documentation as discussed above with respect to the rejection under 35 U.S.C. § 102(b). As discussed above, TETware teaches the features of claim 1, including the claimed test hierarchy.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Eric B. Kiss

Patent Examiner, Art Unit 2192

Conferees:

Tuan Dam

Supervisory Patent Examiner, Art Unit 2192

TUAN DAM
SUPERVISORY PATENT EXAMINER

Eddie Lee

Supervisory Patent Examiner, Technology Center 2100

EDDIE C. LEE
SUPERVISORY PATENT EXAMINER